**BASE DE DATOS I**

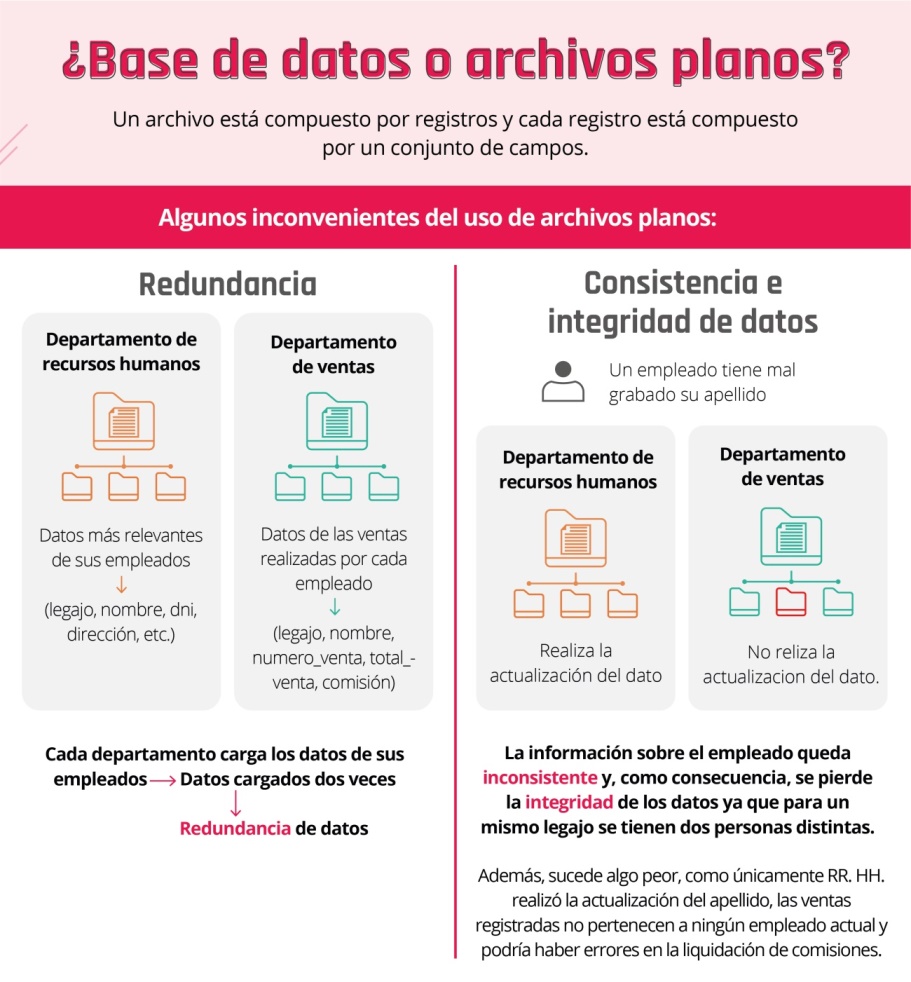
¿Qué es una base de datos? Podemos decir que es un conjunto de datos pertenecientes a un mismo contexto organizados para un propósito específico.

Una base de datos nos permite:

* Almacenar (agregar, modificar y eliminar) datos.
* Acceder a los datos.
* Manipularlos y combinarlos.
* Analizar datos.
* Entre otras cosas más.

Bases de datos las hay de dos tipos: Relacionales y no relacionales. En lo particular, nosotros aprenderemos acerca de una base de datos relacional, ya que la misma nos permite comprender a profundidad cómo funcionan las bases de datos.

**BASES DE DATOS VS. ARCHIVOS PLANOS**



**MOTORES DE BASES DE DATOS**

También conocido como sistema de gestión de base de datos, es una capa de software que almacena, procesar y protege los datos.

Capacidad de los motores de bases de datos:

* Consistencia de los datos
* Seguridad
* Integridad
* Respaldo
* Acceso recurrente
* Tiempo de respuesta

Ejemplos de motores en el mercado:

* Microsoft SQL Server
* My SQL
* IBM DB2
* ORACLE

Todos comparten un estándar en común, por lo que podemos trabajar con una base de datos independientemente de su motor. Ese estándar es SQL, que es un lenguaje que nos asegura el acceso a cualquiera de los motores disponibles en forma transparente. Con este lenguaje se crean bases de datos, se pueden modificar, escribir datos, y consultarlos, leerlos y analizarlos.

**MODELOS DE BASES DE DATOS**

Una base de datos normalmente se comienza a desarrollar con un requerimiento o necesidad de una aplicación o sistema. Lo más importante es saber qué se tiene que almacenar, y detallar este requerimiento lo más específico posible.

Luego de planeado el requerimiento, se pasa al modelado.

Una vez implementada la base de datos, el documento no se elimina, ya que sirve para entender porque se implementó de determinada manera.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Modelado de datos: el modelado de datos es una manera de estructurar y organizar datos para que se puedan usar fácilmente por las bases de datos.

Beneficios:

- Permite observar patrones

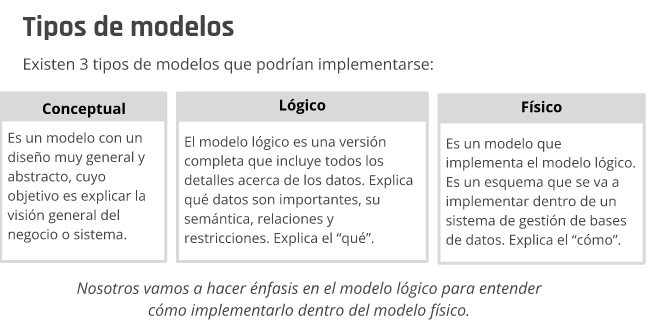
- Se puede descomponer un proceso complejo en partes

- Sirve de plano para construir la base de datos física

- Proporciona soporte ante los cambios de requerimientos de aplicaciones o negocio.

- registrar los requerimientos de datos de un proceso de negocio

Tipos de Modelos:



**MODELO LOGICO VS MODELO FISICO**

Modelo de datos Lógico

Este modelo describe los datos con el mayor detalle posible, indistinto de cómo se implementaran físicamente en la base de datos. Describe los elementos importantes del negocio.

Características:

* Se definen los conceptos importantes sobre los que hay que almacenar información. Estos se denominan **Entidades**.
* Se especifican todos los atributos para cada una de las entidades
* Se conectan las entidades mediante relaciones.
* Se especifica la clave principal para cada entidad
* Se especifican las claves externas (claves que identifican la relación entre diferentes entidades).
* La normalización ocurre en este nivel

Modelo de datos Físico

Este modelo representa **cómo** se construirá el modelo en la base de datos, y muestra todas las estructuras de tablas, nombre de columna, tipo de datos de columna, las restricciones, la clave principal, la externa y la relación entre tablas.

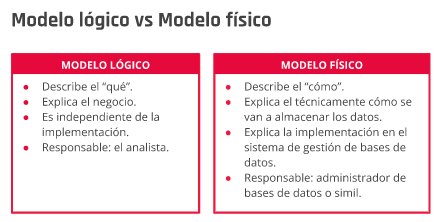
Características:

* Especificación de todas las tablas y columnas
* Las claves externas se usan para identificar relaciones entre tablas
* La des normalización puede ocurrir según los requisitos del usuario

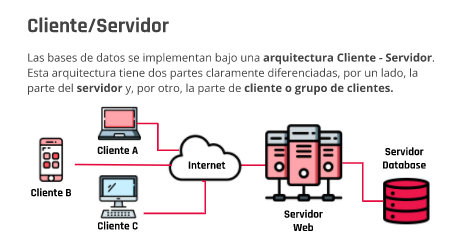
El modelo físico puede diferir de un motor de bases de datos a otro. No es lo mismo implementar en MS SQL Server que en MySQL

Pasos para el diseño del modelo de datos físico:

* Convertir entidades en tablas
* Convertir relaciones en claves externas
* Convertir atributos en columnas
* Modificar el modelo de datos físicos en función de las restricciones/requisitos físicos.



**CLIENTE / SERVIDOR**

****

Ejemplos en la arquitectura:

Cliente: Puede ser MySQL Workbench, o cualquier aplicación que interactúe con la base de datos.

Servidor de base de datos: Va a ser el motor de bases de datos. Ej. MySQL Server

**MODELADO DE BASES DE DATOS**

Con el modelado, no solo vamos a plasmar los requerimientos de aplicaciones sino que también vamos a traducir las reglas o restricciones que deben cumplir los datos.

Tipos de modelos:

* **Modelo conceptual basado en objetos**: Fue definido por Peter Chen en 1976. Se utiliza para la representación de la realidad No comprometida con ningún entorno informático: Sería el Modelo Entidad-Relación propiamente dicho.
* **Modelo lógico basado en objetos**: determinan algunos criterios de almacenamiento y de operaciones de manipulación de los datos dentro de un entorno informático.

**Modelo Entidad – Relación**

El modelo de datos entidad-relación se basa en una percepción del mundo real, que consiste en un conjunto de objetos básicos llamados entidades y de relaciones entre ellos. Se emplea para interpretar, especificar y documentar los requerimientos para los sistemas de bases de datos. Por lo tanto un modelo entidad-relación es un método de representación abstracta del mundo real centrado en las restricciones o propiedades lógicas de una bases de datos.

**Entidades** 🡪 Una entidad es un objeto, real o abstracto, acerca del cual se recoge información de interés para la base de datos.

Tipos de entidades:

* Entidades fuertes: tienen existencia por sí mismas. (alumnos, empleados, departamento.)
* Entidades débiles: dependen de otra entidad para su existencia (hijos de empleados).

Se define como ocurrencia de entidad al conjunto de datos para una entidad en particular. Por ejemplo: 125, Juan Pérez, casado, 23 años. Cada entidad tiene propiedades particulares llamadas atributos.

**Atributos** 🡪 Los atributos describen las características de una entidad. Por ejemplo: Entidad: clientes Atributos: legajo, nombre, domicilio, etc.

Tipos de atributos:

* Atributo con simple valor: Cuando un atributo tiene un simple valor para una identidad particular. Por ejemplo: Una persona que tiene un valor por su fecha de nacimiento y la fecha de nacimiento es un simple valor de la persona.
* Atributo multivalor: Cuando un atributo tiene una serie de valores para identificarse. Por ejemplo: El atributo teléfonos de un cliente que puede contener uno o más números de teléfono.
* Atributos derivados: Cuando los valores de un atributo son afines y el valor para este tipo de atributo se puede derivar de los valores de otros atributos. Por ejemplo: La edad y fecha de nacimiento de una persona; si conocemos la fecha de nacimiento, podemos calcular su edad, en este caso se dice que la edad es un atributo derivado del atributo fecha de nacimiento.
* Atributo clave: Las entidades pueden contener un atributo que identifica cada una de las ocurrencias de la entidad. Es decir, usualmente contienen un atributo que diferencia los ítems entre sí. Por ejemplo: En la entidad clientes el atributo documento puede ser un atributo clave. No necesariamente el atributo clave debe ser un solo atributo, hay casos en que varios atributos forman una llave. Por ejemplo: tipo más número de factura.
* Atributos nulos: Se usa cuando una entidad no tiene valor para un atributo o que el valor es desconocido

Convención de nombres: En los nombres de entidades y atributos siempre se debe utilizar sustantivos en singular o plural. No se puede utilizar eñes, espacios ni acentos. Si el nombre se compone por más de una palabra, se deben reemplazar los espacios con guiones bajos (Snake case) o eliminar dicho espacio y colocar una mayúscula en la inicial de cada palabra (camel case).

**Datos**: Los datos son los posibles valores que pueden tener los atributos. Nos permiten entender si son datos numéricos, textos, fechas, si tienen un formato en particular, si son obligatorios u opcionales. No se modelan, pero nos permite entender mejor las entidades.

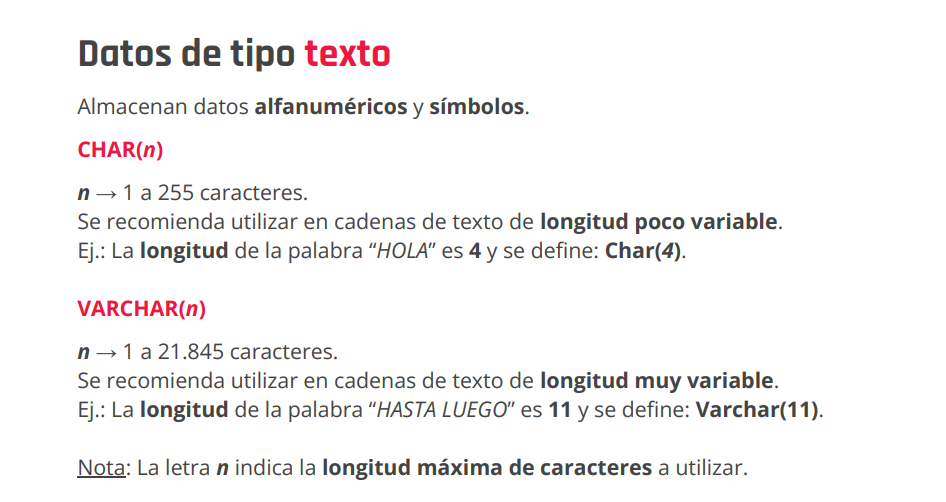
**Claves**

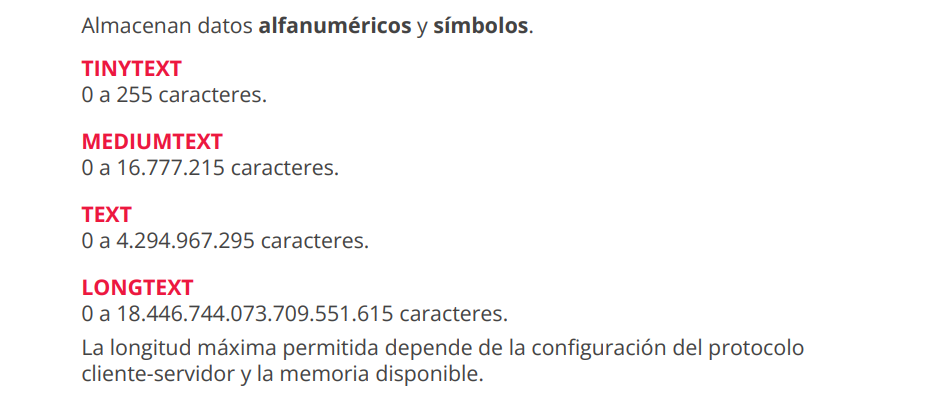
Tipos de claves:

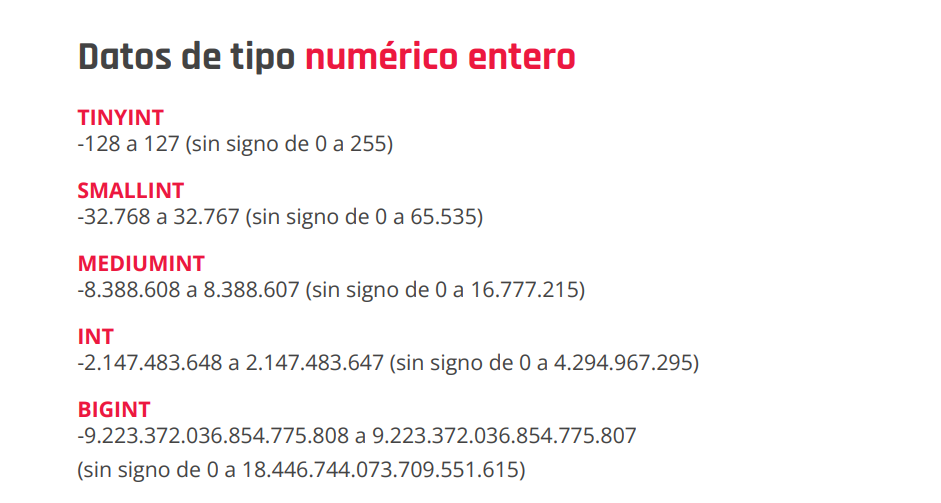
* **Clave candidata**: Se compone por uno o más atributos cuyos valores identifican unívocamente a cada ocurrencia de la entidad, sin que ningún subconjunto de ellos pueda realizar esta misma función. Una clave candidata es una posible clave primaria. Pueden definirse varias claves candidatas para luego seleccionar la más adecuada.
* **Clave primaria**: Está compuesta por uno o más atributos cuyos valores identifican unívocamente a cada ocurrencia de la entidad. No pueden contener valores nulos ni repetidos. Esta clave es una de aquellas que anteriormente se seleccionaron como candidata.
* **Superclave**: es el conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma unívoca a la ocurrencia de una entidad. Se utiliza generalmente en las tablas de relación, este concepto se desarrollará en las próximas clases.

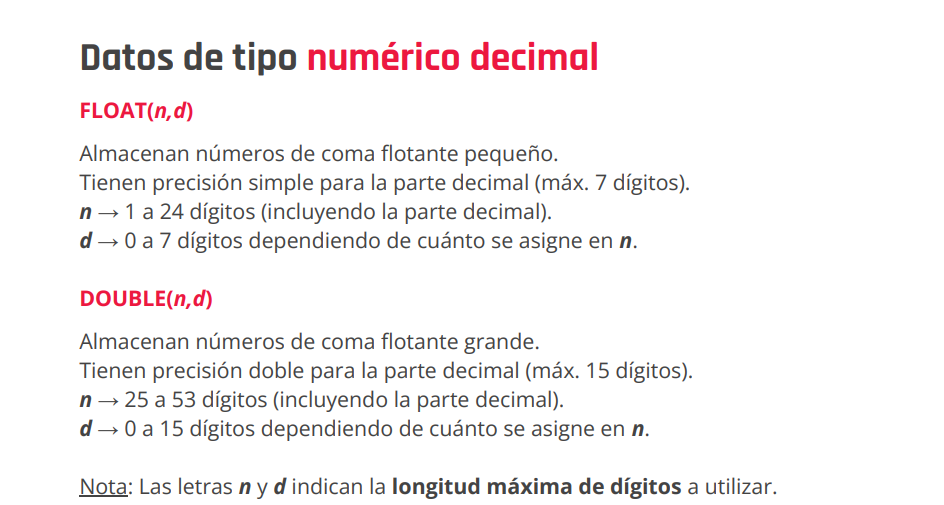
**TIPOS DE DATOS**

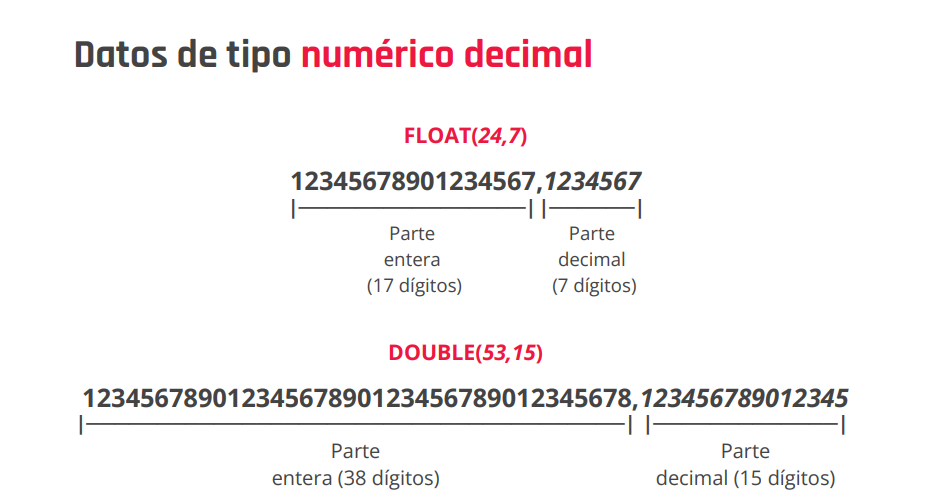
Los datos o atributos de cada registro de una tabla tienen que ser de un tipo de dato concreto. Cuando diseñamos una base de datos tenemos que pensar qué tipo de datos requerimos para nuestro modelo. Cada tipo de dato tiene un tamaño determinado y cuanta más precisión apliquemos en su definición, más rápido y performante va a funcionar MySQL.

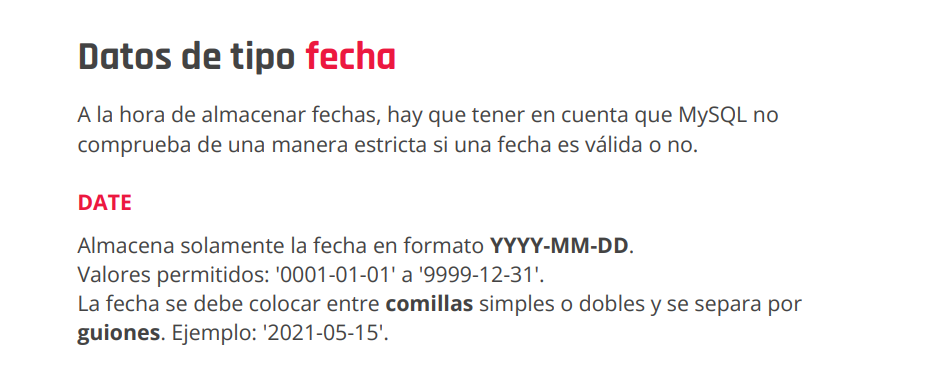


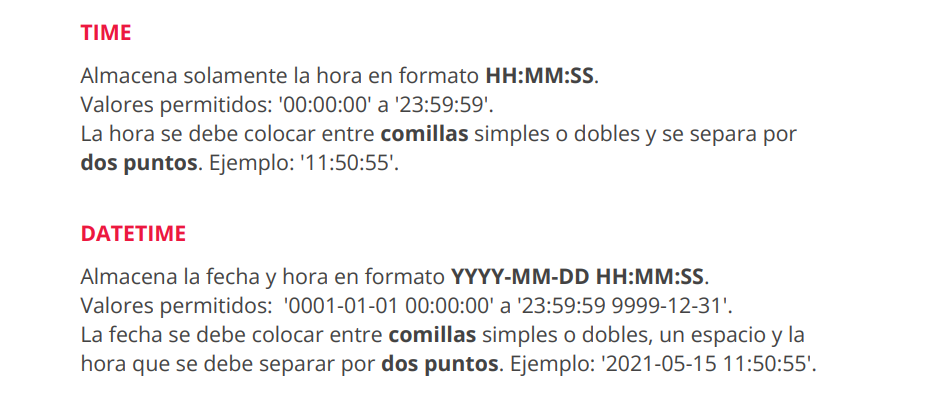


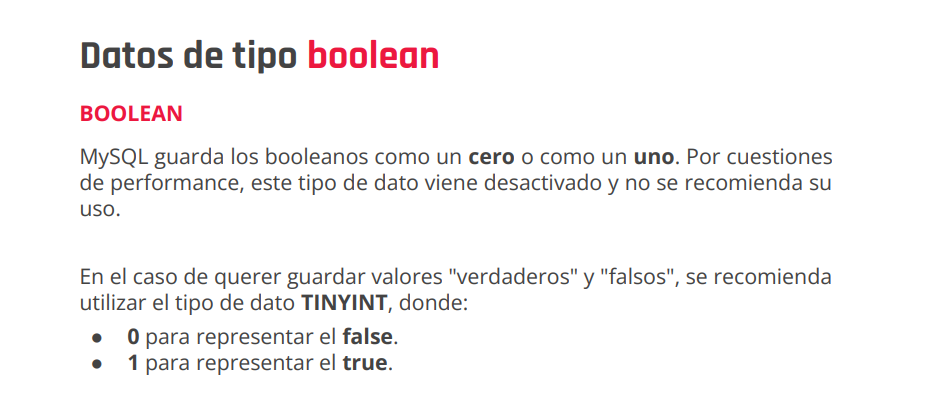












**GLOSARIO DE DATOS**



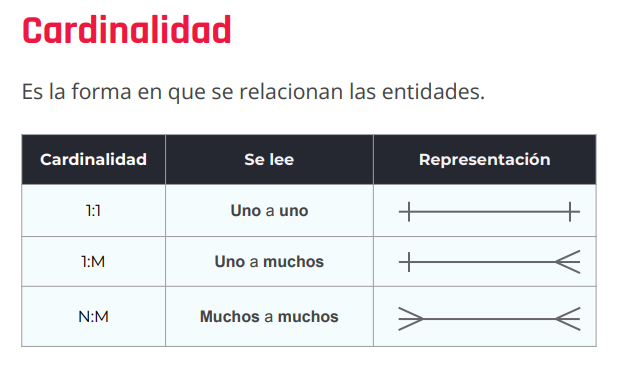
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

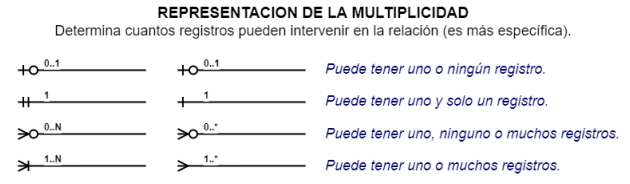
**RELACIONES**

Las relaciones indican cómo se van a relacionar dos tablas. Dentro de una base de datos existen 3 tipos de relaciones:

* Uno a uno: Un usuario tiene solo una dirección. Una dirección pertenece solo a un usuario. Para establecer la relación colocamos la clave primaria de la dirección en la tabla de usuarios, indicando que esa dirección está asociada a ese usuario (Clave foránea).
* Uno a muchos: Un cliente puede tener muchas tarjetas. Una tarjeta pertenece solo a un cliente. Para establecer la relación colocamos la clave primaria del cliente en la tabla de tarjetas, indicando que esas tarjetas están asociadas a un usuario en particular.
* Muchos a muchos: Un cliente puede comprar muchos productos. Un producto puede ser comprado por muchos clientes. En las relaciones N:M, en la base de datos, la relación en sí pasa a ser una tabla. Esta tabla intermedia —también conocida como tabla pívot— puede tener 3 datos: una clave primaria (PK) y dos claves foráneas (FK), cada una haciendo referencia a cada tabla de la relación.

¿Cómo podemos saber cómo se relaciona una entidad con otra? Planteando un ejemplo concreto que nos ayude a definir cómo interactúan esas dos entidades entre sí.





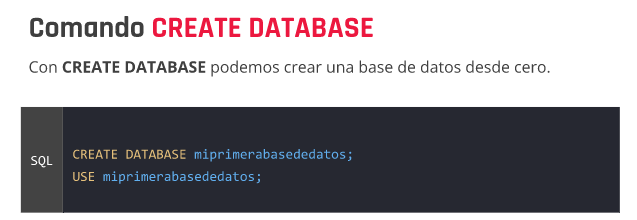
**SQL**

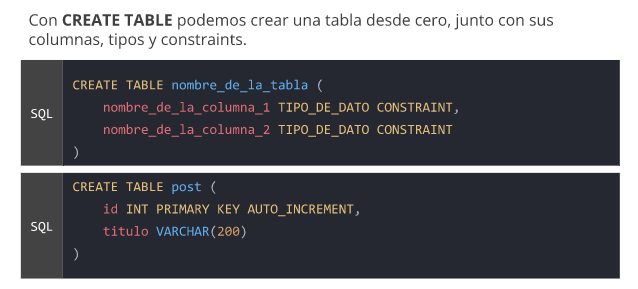
Aprenderemos el lenguaje de consultas SQL

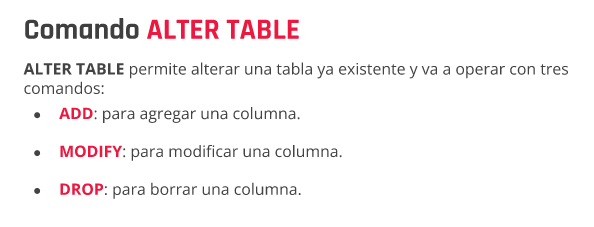
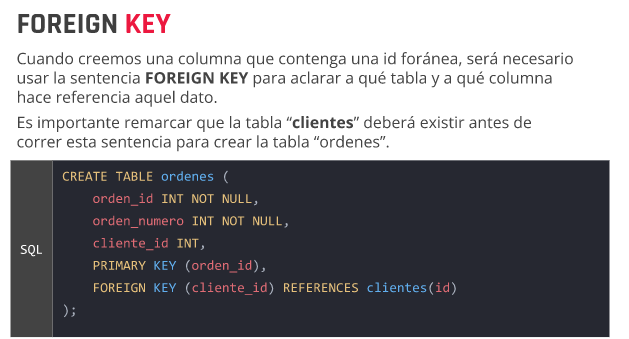
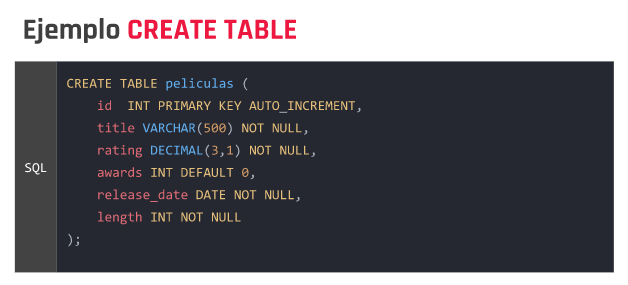
Las sentencias SQL se agrupan en dos categorías según su funcionalidad o propósito:

* **Lenguaje de definición de datos (DDL):** son sentencias para la creación de tablas y registros. Es decir, se utilizan para realizar modificaciones sobre la estructura de la base de datos.
* **Lenguaje de manipulación de datos (DML):** son sentencias para la consulta, actualización y borrado de datos. Es decir, se utilizan para realizar consultas y modificaciones sobre los registros almacenados dentro de cada una de las tablas.

**TEMA: CREATE, DROP, ALTER**: estas directrices nos permiten crear, eliminar, modificar tablas.

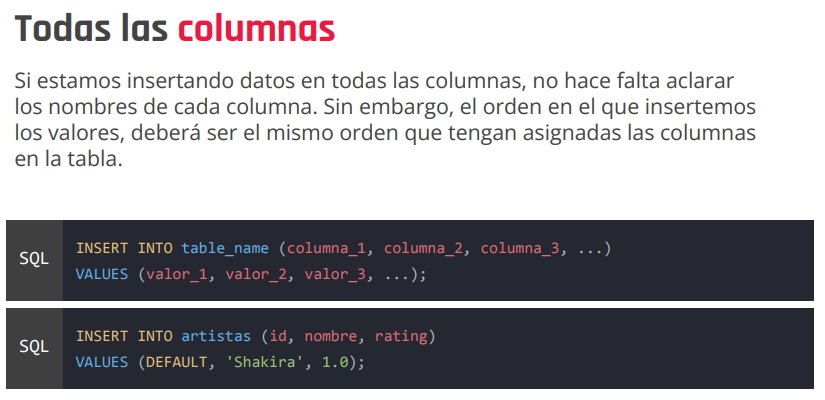
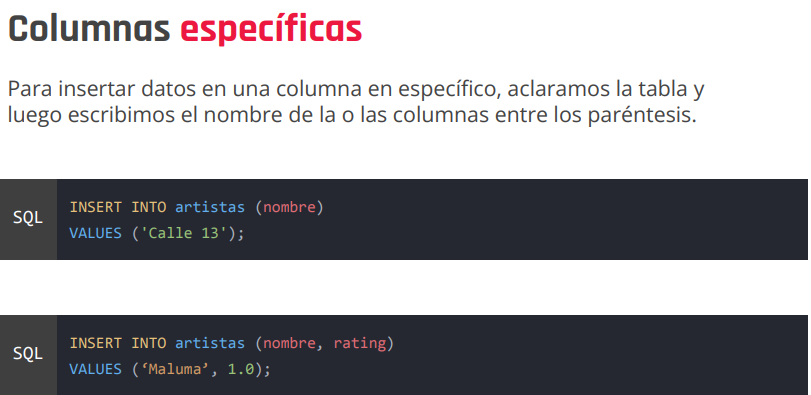




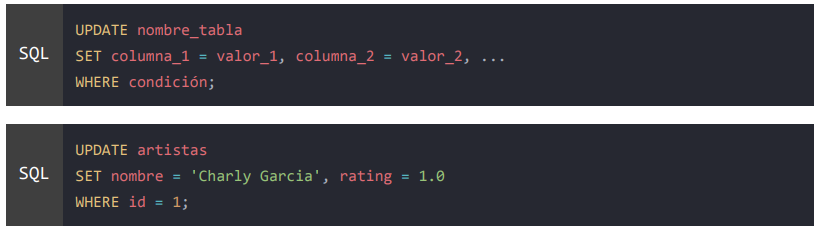


**TEMA: INSERT, UPDATE Y DELETE**. Estas funciones se usan para insertar, actualizar y eliminar registros. **Insert** nos va a permitir agregar datos, con **delete** podremos borrarlos y con **update** podremos modificar los registros existentes en una tabla.

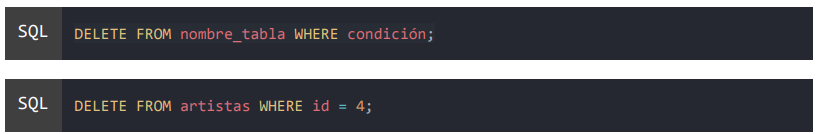
**INSERT** 🡪 Existen dos formas de agregar datos en una tabla:

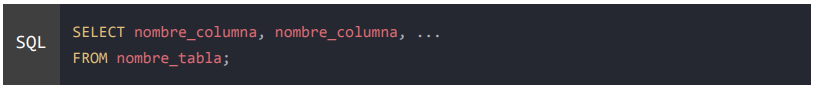
* Insertando datos en todas las columnas.
* Insertando datos en las columnas que especifiquemos.
*  

**UPDATE** 🡪 UPDATE modificará los registros existentes de una tabla. Al igual que con DELETE, es importante no olvidar el WHERE cuando escribimos la sentencia, aclarando la condición.



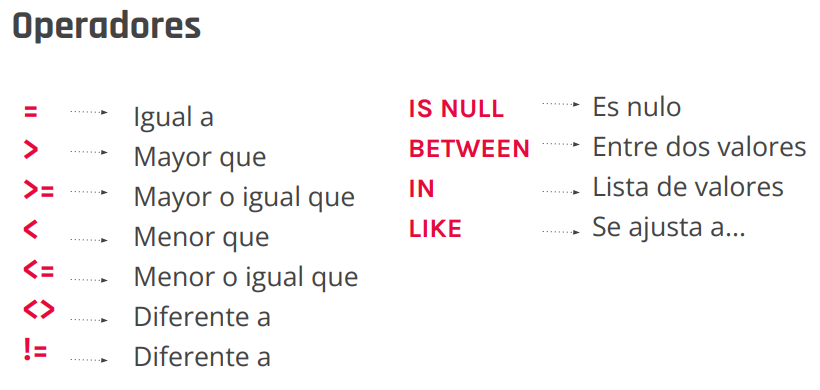
**DELETE** 🡪 Con DELETE podemos borrar información de una tabla. Es importante recordar utilizar siempre el WHERE en la sentencia para agregar la condición de cuáles son las filas que queremos eliminar. Si no escribimos el WHERE, estaríamos borrando toda la tabla y no un registro en particular.



**SELECT** 🡪 Toda consulta a la base de datos va a empezar con la palabra SELECT. Su funcionalidad es la de realizar consultas sobre una o varias columnas de una tabla. Para especificar sobre qué tabla queremos realizar esa consulta usamos la palabra FROM seguida del nombre de la tabla. 

**WHERE y ORDER BY 🡪** La funcionalidad del WHERE es la de condicionar y filtrar las consultas SELECT que se realizan a una base de datos.





**ORDER BY** 🡪 se utiliza para ordenar los resultados de una consulta según el valor de la columna especificada. Por defecto, se ordena de forma ascendente (ASC) según los valores de la columna. También se puede ordenar de manera descendente (DESC) aclarándolo en la consulta.

